Building Segment-Based Maps without Pose Information

Francesco Amigoni, Simone Gasparini, and Maria Gini

***Abstract*— Most map building methods employed by mobile robots are based on the assumption that an estimate of robot poses can be obtained from odometry readings or from observing landmarks or other robots. In this paper we propose methods to build a global geometric map by integrating scans collected by laser range scanners without using any knowledge about the robots poses. We consider scans that are collections of line segments. Our approach increases the flexibility in data collection, since robots do not need to see each other during mapping, and data can be collected by multiple robots or a single robot in one or multiple sessions. Experimental results show the effectiveness of our approach in different types of indoor environments.**

***Index Terms*— Map building, multirobot systems, scan matching, map merging, laser range scanners.**

I. Introduction

Several methods for allowing mobile robots to build maps of unknown environments have been proposed. To build a map, a robot incrementally integrates newly acquired sensor data within previously collected information using knowledge about its own pose or the path it followed since the last integration. Odometry and kinematic models of motion are used to estimate the robot pose (i.e., its position and orientation). Different types of sensors (sonars, laser range scanners, and panoramic cameras) have been used to collect information about the environment. Laser range scanners have become the sensor of choice because of their accuracy and wide availability.

In the last few years, as detailed in the extensive survey by Thrun [1], most of the methods developed for mapping have been based on probabilistic techniques. The methods have in common the fact they use a Bayes filter to recursively compute the posterior probability over robot poses and maps, given the previous sensor measurements and motion commands. The methods differ in the assumptions they make and in how they compute the posterior probability. Some methods operate online, others require multiple passes through the data and so are used offline.

The approach we present in this paper differs from the methods mentioned above in the sense that we do not assume any knowledge of robot pose and we use exclusively range data to construct a bidimensional geometric map composed of line segments. In particular, we show how to build a global map of an environment by merging the post-processed results of the measurement operations performed by laser range scanners, which we call *scans*, without using any position information but relying only on geometric information in the scans. Our approach is similar in spirit to early work by Chatila and Laumond [2] who used geometric descriptions of environments. The advantage of using geometric descriptions over the more common grid-based representations is that line segments can be represented with few numbers and produce maps that are easier to use, as recently discussed for example in [3]. Line segments are also easy to extract automatically from range data.

One might wonder why we do not use odometry, considering that odometric information is often available. The major reason is that we want to use multiple robots to build maps and we want to be able to interrupt the mapping process and resume it at a later time without having to reset the initial poses of the robots (this provides a solution to the so-called "kidnapped robot" problem [4]). In addition, we are interested in building maps with miniature robots, such as the robots described in [5], where no odometry is available. Even if odometry can be replaced in part by sensing (see, for instance, [6]), we believe it is important to understand the implications of not having odometry (which is often unreliable) and to explore what are acceptable bounds on the error on the initial pose of the robots.

In the following, we call *scan* a collection of line segments obtained, as explained later in Section III-D, from the points returned by a 2D laser range scanner. Line segments approximate the points returned by laser range scanners. More precisely, a line segment is represented by its end points (extremes) (x_1, y_1) and (x_2, y_2) in the reference frame of the map. We call *partial map* the result of the integration of two scans, of a scan and a partial map, or of two partial maps. Thus, in the terminology used in this paper, a scan is a special case of a partial map. Both scans and partial maps are collection of line segments. The difference is that we assume the line segments in a scan are ordered (clockwise or counterclockwise), while they are not in a partial map. Range data can be collected by a single or different robots. We assume that the robots move indoor on a 2D surface and that walls and vertical objects are at the height of the laser scan. No other assumption is made about the environment to be mapped: the environment is supposed to be unknown and with no ground-truth maps available. Experiments demonstrate that our method works both in regular and in scattered environments.

This paper presents two main contributions. The first is a method for *integrating two partial maps* (and, in particular, two scans) relying exclusively on their geometry. We consider

Francesco Amigoni and Simone Gasparini are with the Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Milano, Italy. Maria Gini is with the Department of Computer Science and Engineering of the University of Minnesota, Minneapolis, USA.

the angles between pairs of line segments in the maps as a sort of "geometric landmarks" [7] on which our matching process is based: the idea is to match angles of the two partial maps that are similar. This method is robust to large displacements between the partial maps, provided that the maps have an overlap containing at least an angle representing the same portion of the environment. The method integrates two partial maps, S_1 and S_2, into a map $S_{1,2}$ in three major steps:

1) *find the possible transformations* of S_2 on S_1;
2) *evaluate the transformations* to identify the best transformation $\bar{t}$ of S_2 on S_1;
3) *apply the best transformation* $\bar{t}$ to S_2 (obtaining $S_2^{\bar{t}}$) and *fuse the line segments* of S_1 and $S_2^{\bar{t}}$ to obtain $S_{1,2}$.

When S_1 and S_2 are two scans, the first two steps above are known in literature as *scan matching*.

The second main contribution of this paper is the proposal of three methods for *map merging* that integrate a *sequence* $S_1, S_2, \ldots S_n$ of n partial maps. The sequence defines the order in which the partial maps must be integrated; namely S_1 has to be integrated with S_2, that in turn has to be integrated with S_3, and so on. We reduce the merging of a sequence of partial maps to the iterated integration of two partial maps.

Our approach for scan matching and map merging without pose information has the advantage of being independent from how the data have been collected. It is indifferent if the scans are collected during a single session or multiple sessions, by multiple robots or a single robot. Robots can be added or removed at any time, and they do not need to know their own position. For the experiments in this paper we used scans acquired by a single robot but all the results are applicable to multirobots. In this case, the mapping process is centralized, with data collected to a location and assembled in a map. More precisely, in a multirobot scenario, our method can be applied at two levels: at the individual robot level, the method integrates the sequence of scans acquired by the robot and, at the global system level, the method integrates, less frequently, the partial maps built by the robots.

The paper is structured as follows. In the next section we outline previous work on scan matching and map merging and compare it with our approach. We present our scan matching method in Section III, and we illustrate our map merging method in Section IV. Section V covers the experimental validation of our contributions. Section VI concludes the paper.

II. Related Work

Robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots [1]. These spatial models, or *maps*, are typically used for robot navigation, for example to plan paths. In order to build a map, robots use sensors like sonars, cameras, and laser range scanners. The range limitations of these sensors force the robots to navigate in the environment while building the map. As a consequence, maps are built incrementally, integrating the newly perceived information within the already available map. Usually, this integration exploits the (uncertain) knowledge about robot poses. Maps can be represented topologically (e.g., by graph-based data structures) or geometrically (e.g., by data structures storing grids, points, or line segments). Since grid-based maps require high-dimensional representations with thousands of numbers, segment-based maps have been recently advocated to reduce dimensionality [3].

Multirobot mapping has attracted attention in recent years because of the robustness of exploring in parallel with multiple robots and of the potential savings in the time needed to map large areas. A key challenge in multirobot mapping is merging the maps produced by independent robots. The methods proposed in the literature to address the *map merging* problem have been mostly based on building robot centric maps and merging them using the relative positions of the robots, which must be known.

One way of doing this is to extend SLAM (Simultaneous Localization and Mapping) or CML (Concurrent Mapping and Localization) techniques [8], [9], [10] to multirobots. SLAM and CML techniques are widely employed in the context of single robot mapping and refer to the problem of building a map and, at the same time, estimating the pose of the robot. Since odometric measurements are noisy [11], robot localization cannot rely only on dead-reckoning, and a probabilistic machinery is employed to localize the robot in the map that is being constructed. A family of approaches adopts Kalman filtering [12], [13] in an incremental process that estimates robot pose and landmarks' positions in the map. This solution requires a large computational effort as the number of features in the map grows and it is also not well suited to dynamic environments and environments with indistinguishable landmarks. Another probabilistic approach is based on the Expectation-Maximization (EM) algorithm [14], [15], [16], [17] and is usually employed to build grid-based maps. Robot pose is tracked by a multimodal probability density function which copes well with the correspondence problem (i.e., having to associate sensor data with features in the map) and with failure recovery. To alleviate the computational burden, faster methods have been developed, including particle filters [9] and FastSLAM [18].

Most map merging techniques rely on the assumption that the robot poses are known. For example, in [8], [14] the pose of the robots is assumed to be known at all times; in [16] the robots don't know their relative starting positions but each robot has to start within sight of the team leader; in [17] the robots must start in known nearby locations; in [19] the robots have to see each other from time to time. In [20] a stationary robot continuously tracks the motion of another robot which acquires sensory data from the environment. In [9], particle filters are used for partial map localization. The robots have to actively verify their relative poses before the maps are merged, and the integration of partial maps is not fully automated. In [21] a single robot is used with FastSLAM to generate maps directly from laser range data. The method aligns a scan to the previous one by computing an occupancy grid map [22]. The method, which requires a model of the odometry error, is robust and converges even in cases where the standard Rao-Blackwellized particle filter [23] without odometry correction fails to converge.

An exception is the work reported in [24] where map merging is done using a decision theoretic approach. The robots do not need to know their own position, but the maps have to be annotated with distinctive features. This step is currently done manually. The match is done not with individual scans but with patches made of 15 scans taken $0.5\,\mathrm{m}$ apart, each of them containing 2 to 8 distinctive features. This is an improvement over earlier work [25], where map merging was done by correlation of a patch over a partial map. The method required the scans to be taken close to each other ($30\,\mathrm{cm}$ in the examples shown), and a very good scan matching algorithm, since map merging could not be undone.

The approach we present in this paper is based on building geometric maps, which are represented as collections of line segments, from scans, which are also collections of line segments. In the literature, scans are either based on line segments or on raw sensor data. *Scan matching* is the process of calculating the translation and rotation of a scan to maximize its overlap with a reference scan.

Lu and Milios [26], [27] introduced the idea of consistent pose registration, where scans from a laser range scanner taken at different poses are matched using a priori information and odometry constraints between successive poses. The algorithm does multiple passes through the data, so it is not real-time. The algorithm iteratively minimizes an error measure by first finding a correspondence between points in the two scans, and then doing a least square minimization of all the point-to-point distances to determine the best transformation. An initial pose estimate is provided through odometry to avoid erroneous alignments. The method works very well when the errors in the initial position are small ($< 20\,\mathrm{cm}$).

Another well-known technique for scan matching is the iterative algorithm of Cox [28] for matching range scans to an *a priori* map of line segments. Since it assumes small displacements between a scan and the map, the algorithm first finds the correspondence between scan points and line segments and then calculates the translation and rotation that minimize the (square of all) point-to-segment distances. The two steps are repeated until the process converges. Each iteration returns a position correction vector and a variance-covariance matrix that evaluates the match. This approach has been extended in [29], where line segments are extracted from the previous scans and used as the reference model for matching, instead of using an *a priori* model of the environment, These methods can be applied only to polygonal environments, a limitation that our method tries to overcome.

With straight perpendicular walls, matching can be done using histograms, as in [30], where the orientation is computed by cross-correlation of the histograms of the angles between the actual and previous scans, and the translation by cross-correlation of the distance histogram. This method is sensitive to large displacements between the maps and to changes in the environment. The improvement proposed in [31] deals with non-perpendicular walls and segment maps, even if it still assumes straight walls and has poor performance in scattered environments.

In [32] line segments are extracted from range points and a special “center of gravity” representation is used to describe the uncertainty of line segments. Pairs of line segments are matched and the translation is computed by least square minimization, using an initial estimate for the displacement provided by odometry. The technique proposed in [33] refines the alignment of scans collected by multiple robots using the partial Hausdorff distance to compute the best transformation between a new scan and a point map of the previously explored region. This method also requires an initial estimate of the pose of the scans. The method proposed in [34] is similar to our approach in that it extracts line segments from laser range scanner readings and builds incrementally a global map. It first determines the relative orientation of the two maps by computing the histogram of the angle differences and then adjusts the translation by overlapping the line segments using least square minimization. The method works for linear and static environments and for very small displacements. Our method is more general and allows for significant displacements between two partial maps, provided that they have at least a corresponding angle representing the same portion of the environment.

There have been a few attempts to match scans independently of odometry. For example, [35] proposes to use a panoramic range finder to build segment maps. It identifies line segments representing walls or other boundaries of the environment and matches the scans taken from different positions without relying on any additional source of information. This is accomplished by applying a dynamic programming algorithm to the vertical lines of the map. The method operates in polygonal or rectilinear environments, but does not work well in scattered environments and it (implicitly) relies on small displacements of the robot. In [36], segment maps are matched by establishing a correspondence between their features. This is reduced to measure shape similarity between polylines according to their maximal convex arcs, following a method originated in computer vision. Although no data are reported about the displacement of the maps, from the reported experiments it can be inferred to be around $0.5\,\mathrm{m}$.

In [37], a scan matching algorithm extending geometric hashing is proposed. The main idea is a signature representation of the local region around each point of the scan. The search for the best alignment between two scans is performed with a voting system in the Hough space containing all the signatures. The candidate alignment is then applied to the measurement model of the SLAM framework. From the reported experimental data, it seems that the system is implicitly based on small displacement between two scans, about $20 - 30\,\mathrm{cm}$. The scan matching method proposed in [38] does not use information about odometry to compute the alignment between two scans. It is based on geometric features of the maps, the so-called Complete Line Segment relationships. All the line segments that completely represent real objects in the environment are singled out and their relative position is used to find a correspondence with the Complete Line Segments of the old map. The method has been shown to be fast and accurate, even if it is not clear how it can deal with ambiguity and the case of non-occluded but partially visible line segments (features that go beyond the visibility region of the sensor). It also seems that it cannot be extended to multirobot map building with unknown position of the robots, since it is weakly based on a sorting order of the line segments to improve search efficiency.

III. Method for Scan Integration

In this section, we present our method for integrating two scans. The method works in the three steps outlined in Section I. Our algorithm INTEGRATE (reported as Algorithm 1) is exclusively based on the geometric information and constraints [7] contained in the scans. In particular, we consider angles between pairs of line segments in the scans as a sort of "geometric landmarks" on which the matching process is based. This use of "local" geometric features is significantly different from other related works in map building that use "global" geometric features (e.g., those represented by an histogram of angle differences).

INTEGRATE integrates two scans into a partial map. Let's call S_1 and S_2 the two scans and $S_{1,2}$ the resulting partial map. Although in the following we discuss, for simplicity, the integration of two scans, all the methods are applicable to the integration of two partial maps. In the algorithms below, two points are considered to coincide when they are closer than POINTDISTANCETOLERANCE (in our experiments we set this parameter to $15\,\mathrm{mm}$) and two angles are considered equal when their values differ of less than ANGLEDIFFERENCETOLERANCE (in our experiments we set this parameter to $0.2\,\mathrm{rad}$).

Algorithm 1 INTEGRATE

```
   Input: two scans $S_1$ and $S_2$
   Output: a map $S_{1,2}$
1: $T \leftarrow$ all transformations of $S_2$ on $S_1$     ▷ see Algorithm 2
2: $\bar{t} \leftarrow$ best transformation in $T$          ▷ see Algorithm 4
3: $S_{1,2} \leftarrow$ fusion of $S_1$ and $S_2^{\bar{t}}$       ▷ see Algorithm 5
```

Algorithm 2 Find transformations

```
    Input: two scans $S_1$ and $S_2$
    Output: a set of transformations $T$
 1: $A_1 \leftarrow$ empty
 2: for all pairs of line segments $s_1$ and $s'_1$ ($s_1 \neq s'_1$) in $S_1$ do
 3:     add to $A_1$ the angle between $s_1$ and $s'_1$
 4: end for
 5: $A_2 \leftarrow$ empty
 6: for all pairs of line segments $s_2$ and $s'_2$ ($s_2 \neq s'_2$) in $S_2$ do
 7:     add to $A_2$ the angle between $s_2$ and $s'_2$
 8: end for
 9: $T \leftarrow$ empty
10: for all pairs of angles $\alpha_1 \in A_1$ and $\alpha_2 \in A_2$ do
11:     if $\alpha_1 = \alpha_2$ then
12:         add to $T$ the rototranslation that superimposes $\alpha_2$ to $\alpha_1$
13:     end if
14: end for
```

A. Find Transformations

This step, given the scans S_1 and S_2, first finds the angles between the line segments in S_1 and between the line segments in S_2 and, second, finds the possible transformations (namely, the rotations and translations) that superimpose at least one angle α_2 of S_2 to an equal angle α_1 of S_1. We use the angles between pairs of line segments as geometric landmarks and we try to match equal angles in the two scans. The pseudo-code is reported as Algorithm 2. Finding the possible transformations is a difficult combinatorial problem since in principle, without any information about the relative poses of the two scans, there are $O(n_1^2 n_2^2)$ possible transformations, where n_1 and n_2 are the numbers of line segments in S_1 and S_2, respectively. We have therefore devised three heuristics for reducing this complexity and finding a set of (hopefully) significant transformations between two scans. They are described in the following.

1) Consider Angles between Consecutive Line Segments in a Scan: In each scan, we select the angles between two consecutive line segments; let A_1^c and A_2^c be the sets of such angles for S_1 and S_2, respectively. Two line segments are consecutive when they have an extreme point in common. Then, we find the set of all the transformations that make an angle in A_2^c to correspond to an equal angle in A_1^c. This amounts to change the step 3 (respectively, 7) of Algorithm 2. The modified step adds an angle to A_1 (A_2) only when s_1 and s'_1 (s_2 and s'_2) are consecutive. The number of possible transformations found by this method is $O(n_1 n_2)$. We note that finding the sets A_1^c and A_2^c is greatly facilitated when the line segments in S_1 and in S_2 are ordered. This is usually the case when scans are acquired with laser range scanners, since the points returned by the sensor are ordered counterclockwise and it is straightforward to maintain the same order in the line segments that approximate the points.

Although this heuristic seems to perform well in indoor environments where consecutive walls are usually perpendicular, the errors introduced by the sensor (for example due to irregular reflection patterns) and by the algorithm that approximates points with line segments may alter the representation of these angles. Hence, the angles between consecutive line segments sometimes do not constitute a good model of the environment angles.

To improve the performance of this heuristic, we can consider angles between consecutive line segments even when the line segments do not have a common extreme point (this could be done only if the line segments of the scans are ordered). Moreover, consecutive line segments can be considered to form a significant angle only if they are longer than a fraction (specified by the parameter SEGMENTLENGTHPERCENTAGE, set to an average value of 20 in our experiments) of the longest line segment in the scan. The implicit assumption is that long line segments are more reliable than short line segments in representing the environment. Although this improvement gives good results with scans, it is not easily applicable to partial maps in which an order on the line segments is often hard to define.

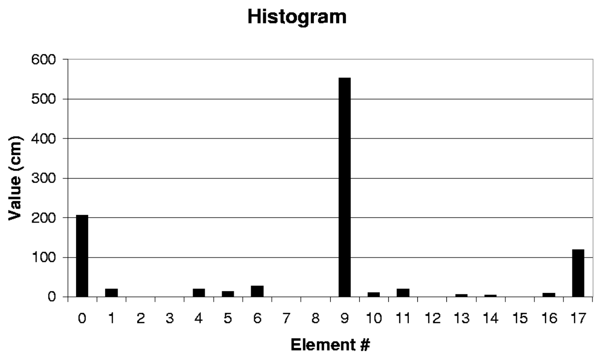
2) Consider Angles between Randomly Selected Line Segments in a Scan: In each scan, we examine a number of angles between pairs of line segments selected randomly. We assign a higher probability to be selected to longer line segments, since they provide more precise information about the environment. Let A_1^r and A_2^r be the sets of the selected angles for S_1 and S_2, respectively. We find the set of all the transformations that bring an angle in A_2^r to correspond to an equal angle in A_1^r. The number of transformations generated by this method is $O(a_1 a_2)$, where $a_1 = |A_1^r|$ and $a_2 = |A_2^r|$ are the number of angles in A_1^r and A_2^r, respectively.

Instead of assigning directly to each line segment the probability of being selected (according to its length) and of selecting a number a_1 (respectively, a_2) of pairs, the following approximate and easy-to-implement technique is employed. Initially only line segments longer than SEGMENTDIVISIONFACTOR (set to 0.5 in our experiments) times the length of the longest line segment in S_1 (respectively, S_2) are considered for selection. All the line segments considered have equal probability of being selected. Then, we proceed to iterate with $k = 1, 2, \ldots, K$. In the k-th iteration, we use a threshold equal to SEGMENTDIVISIONFACTOR^k times the length of the longest line segment in S_1 (respectively, S_2). Out of the line segments longer than this threshold we select one with equal probability. Thus, the parameter SEGMENTDIVISIONFACTOR determines the length of the line segments that are considered for selection and, implicitly, the probability of selection. The pseudo-code of the algorithm is reported as Algorithm 3. This technique tries first to find transformations based on angles between long line segments; then it progressively considers transformations based on angles between shorter and shorter line segments. The above technique can be further improved by stopping the generation of transformations when a "good enough" transformation is found. (The evaluation of the quality of a transformation is discussed in Section III-B.)

3) Consider Angles between Perpendicular Line Segments in a Scan: In each scan, we select only angles between perpendicular line segments. This amounts to change step 3 (respectively, 7) of Algorithm 2. The modified step adds an angle to A_1 (A_2) only when s_1 and s_1' (s_2 and s_2') are perpendicular. This heuristic is particularly convenient for indoor environments, where the presence of regular walls usually involves perpendicular line segments. To make this heuristic more efficient, we used histograms. The *histogram* of S_1 (and, in similar way, that of S_2) has *nslots* buckets. Each bucket L_i ($i = 0, 1, \ldots, nslots - 1$) contains the line segments with orientation comprised between $\pi \times i/nslots$ and $\pi \times (i+1)/nslots$, measured with respect to a given reference axis. To each element L_i of the histogram is associated a value calculated as the sum of the lengths of the line segments in L_i. The *principal direction* of an histogram is the element with maximum value. The *normal direction* of an histogram is the element that is $\pi/2\,\mathrm{rad}$ away from the principal direction. In Fig. 1, the histogram of a scan taken in an indoor environment is shown (with $nslots = 18$). The principal direction is the element L_9 and the normal direction is the element L_0. Let A_1^h and A_2^h be the sets of angles formed by a line segment in the principal direction and by a line segment in the normal direction of the histograms of S_1 and S_2, respectively. The set of possible transformations is then found comparing the angles in A_1^h and A_2^h. The number of possible transformations generated by the above heuristic is $O(p_1 q_1 p_2 q_2)$, where p_i and q_i are the number of line segments in the principal and normal directions of the histogram of scan S_i.

Algorithm 3 Find transformations based on angles between pairs of randomly selected line segments

```
    **Input**: two scans $S_1$ and $S_2$
    **Output**: a set of transformations $T$
 1: $T \leftarrow$ empty
 2: for $k = 1, 2, \ldots, K$ do
 3:     for $i = 1, 2, \ldots, N_k$ do   ▷ $N_k$ has been set to $20 \cdot 2^k$ in our experiments
 4:         pick up randomly two line segments $s_1$ and $s_1'$ ($s_1 \neq s_1'$) from $S_1$ that are longer than SEGMENTDIVISIONFACTOR$^k$ times the length of the longest line segment is $S_1$
 5:         $\alpha_1 \leftarrow$ angle between $s_1$ and $s_1'$
 6:         pick up randomly two line segments $s_2$ and $s_2'$ ($s_2 \neq s_2'$) from $S_2$ that are longer than SEGMENTDIVISIONFACTOR$^k$ times the length of the longest line segment is $S_2$
 7:         $\alpha_2 \leftarrow$ angle between $s_2$ and $s_2'$
 8:         if $\alpha_1 = \alpha_2$ then
 9:             add to $T$ the rototranslation that superimposes $\alpha_2$ to $\alpha_1$
10:         end if
11:     end for
12: end for
```



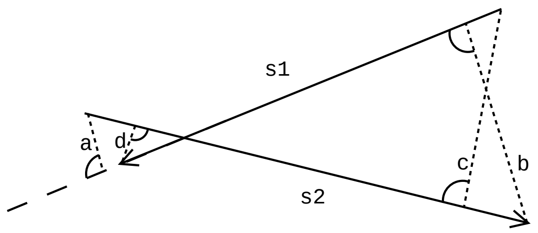
1: The histogram of a scan

B. Evaluate Transformations

Every transformation found in the previous step needs to be evaluated in order to identify the best one. To determine the goodness of a transformation t we transform S_2 on S_1 (in the reference frame of S_1) according to t (obtaining S_2^t), then we calculate the approximate length of the line segments of S_1 that correspond to (namely, match with) line segments of S_2^t. The *transformation value* is the length of the corresponding line segments that the transformation produces. More precisely, the value of a transformation is the sum of all the matching values calculated for every pair of line segments $s_1 \in S_1$ and $s_2^t \in S_2^t$. The *matching value* between two line segments s_1 and s_2^t is calculated as follows. We project s_2^t on the line supporting s_1 thus obtaining a projected line segment s_{2p}^t and then we compute the length l_1 of the common part of s_1 and s_{2p}^t; we do the same but projecting s_1 on s_2^t, obtaining l_2. The matching value of s_1 and s_2^t is calculated as the average of l_1 and l_2. When s_1 and s_2^t do not intersect, the matching value is multiplied by $0.95^{d(s_1,s_2^t)/\textsc{PointDistanceTolerance}}$ to penalize the match between line segments that are far away. Note that 0.95 is an empirical constant whose value has been determined during experimental activities. $d(s_1, s_2^t)$ is the distance between two line segments, calculated as in [7]:

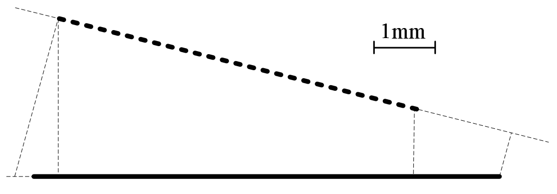
$$d(s_1,s_2) = \min(\max(\text{dist}(s_1,\text{start}(s_2)),\text{dist}(s_1,\text{end}(s_2))),\\ \max(\text{dist}(s_2,\text{start}(s_1)),\text{dist}(s_2,\text{end}(s_1))))$$

where start and end are the extremes of a line segment and dist(s,p) is the Euclidean distance between point p and the line supporting segment s (Fig. 2, note that $s_1 \in S_1$ and $s_2 \in S_2$). Note that, usually, in computer graphics and in computer vision, the distance between two sets of points A and B is calculated as the *Hausdorff distance* $H(A,B) = \max(h(A,B),h(B,A))$, where $h(A,B) = \max_{a\in A}(\min_{b\in B}(||a-b||))$, and $||\cdot||$ is the Euclidean distance. One can show that $d(s_1,s_2) \le H(s_1,s_2)$ for any segment s_1 and s_2. The role of penalization for two pairs of line segments is illustrated in Fig. 3, where line segments belonging to different scans are represented by continuous and dotted lines, respectively. When two line segments have a positive matching value they (supposedly) represent the same part of the environment.

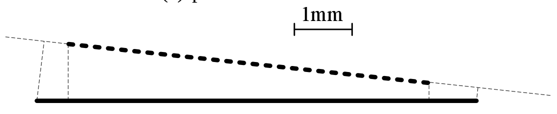


2: The distance between s_1 and s_2 is $d(s_1,s_2) = \min(\max(a,b),\max(c,d))$, where the marked angles are equal to $\pi/2$

Finally, two special cases can appear during the evaluation of the matching value of s_1 and s_2^t. The matching value is set to 0 when the two line segments are too far away, namely when the ratio of $d(s_1,s_2^t)$ to PointDistanceTolerance is larger than SegmentDistanceThreshold. SegmentDistanceThreshold is usually set to 5 to obtain good experimental results. The transformation is discarded when the two line segments intersect and are longer than SegmentLengthRefuse (usually set to 80 cm or 100 cm in our experiments). The pseudo-code of the algorithm is reported as Algorithm 4. (Steps 24-30 are explained in the next section.)



(a) penalization = 0.9150



(b) penalization = 0.9735

3: Different values for the penalization for the pair of line segments shown

Algorithm 4 Find the best transformation

```
Input: two scans, $S_1$ and $S_2$, and a set of transformations $T$
Output: the best transformation $\bar{t}$ in $T$
1: $btv \leftarrow 0$    ▷ current best transformation value
2: for all $t \in T$ do
3:     $S_2^t \leftarrow$ transform $S_2$ according to $t$
4:     $tv \leftarrow 0$    ▷ value of $t$
5:     for all pairs of line segments $s_1 \in S_1$ and $s_2^t \in S_2^t$ do
6:         if $s_1$ and $s_2^t$ intersect and both are longer than SegmentLengthRefuse then
7:             consider the next transformation    ▷ next iteration of the outer for
8:         end if
9:         if $d(s_1,s_2^t)$/PointDistanceTolerance $>$ SegmentDistanceThreshold then
10:            consider the next pair of line segments    ▷ next iteration of the inner for
11:        end if
12:        $s_{2p}^t \leftarrow$ projection of $s_2^t$ on line supporting $s_1$; $s_{1p} \leftarrow$ projection of $s_1$ on line supporting $s_2^t$
13:        $l_1 \leftarrow$ length of common part of $s_1$ and $s_{2p}^t$; $l_2 \leftarrow$ length of common part of $s_2^t$ and $s_{1p}$
14:        $mv \leftarrow (l_1 + l_2)/2$    ▷ matching value
15:        if $s_1$ and $s_2^t$ do not intersect then
16:            $mv \leftarrow mv * 0.95^{d(s_1,s_2^t)/\textsc{PointDistanceTolerance}}$    ▷ penalization
17:        end if
18:        $tv \leftarrow tv + mv$
19:    end for
20:    if $tv > btv$ then
21:        $\bar{t} \leftarrow t$; $btv \leftarrow tv$
22:    end if
23: end for
24: for all pairs of line segments $s_1 \in S_1$ and $s_2^{\bar{t}} \in S_2^{\bar{t}}$ do
25:     if $(s_1,\cdot)$ or $(\cdot,s_2^{\bar{t}})$ already belongs to a matching chain $C$ relative to $\bar{t}$ then
26:         add to $C$ the pair $(s_1,s_2^{\bar{t}})$
27:     else
28:         create a new matching chain relative to $\bar{t}$ and add to it the pair $(s_1,s_2^{\bar{t}})$
29:     end if
30: end for
```

The above algorithm evaluates a single transformation by considering all the pairs of line segments of the two scans that are $O(n_1 n_2)$. A way to limit this computational effort is to stop the evaluation of a transformation t when its value cannot be larger than the current maximum, namely when the length of the line segments of S_1 (or S_2^t) whose matching value has been not yet calculated is less than the difference between the current value of t and the current maximum.

C. Apply the Best Transformation and Fuse the Scans

Once the best transformation $\bar{t}$ has been found, the third and last step of our scan integration method transforms the second scan S_2 in the reference frame of S_1 according to $\bar{t}$, obtaining $S_2^{\bar{t}}$.

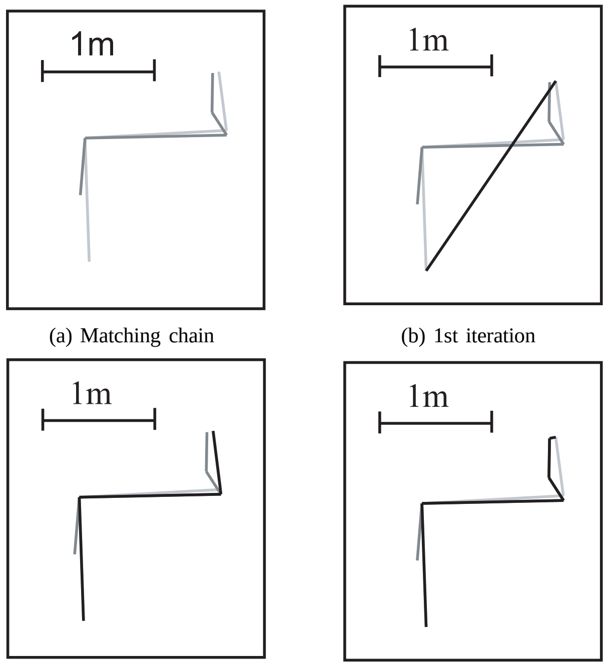
Because of calculation, scan, and matching errors the scans might not align exactly. To produce the output map $S_{1,2}$ we first replace each matching chain created in steps 24-30 of Algorithm 4 with a polyline, and we add the resulting polylines to the unmatched line segments of S_1 and $S_2^{\bar{t}}$.

A *matching chain* relative to transformation $\bar{t}$ for the pair of scans S_1 and $S_2^{\bar{t}}$ is the set $C = \{\langle s_1, s_2^{\bar{t}}\rangle | s_1 \in S_1$ and $s_2^{\bar{t}} \in S_2^{\bar{t}}$ have a positive matching value for $\bar{t}\}$ algebraically closed under line segment belong-to relation. Specifically, a matching chain C is such that if $\langle s_1, s_2^{\bar{t}}\rangle \in C$, then all the line segments s that have a positive matching value (namely, have matched with) s_1 or $s_2^{\bar{t}}$ belong to C; i.e., $\langle s_1, s\rangle \in C$ and $\langle s, s_2^{\bar{t}}\rangle \in C$. We explicitly note that, given an element $\langle s_1, s_2^{\bar{t}}\rangle$, the matching chain C that contains (that is generated by) $\langle s_1, s_2^{\bar{t}}\rangle$ is uniquely identified. It is easy to see that a transformation $\bar{t}$ generates a set of (disjoint) matching chains.

Each matching chain (i.e., each set of pairs of corresponding line segments) is fused in a single polyline, which then replaces the corresponding line segments in the final map. Therefore, the final map is obtained by adding the polylines that represent the matched line segments (i.e. the line segments in the matching chains) to the unmatched line segments of S_1 and $S_2^{\bar{t}}$. The pseudo-code is in Algorithm 5 and an example is shown in Fig. 4.

We build the polyline that approximates the line segments in a matching chain C by iteratively building a sequence of approximating polylines $P_0, P_1, \ldots$ that converges to the polyline P that adequately approximates (and substitutes in the resulting map) the matching line segments in C. The polyline P_0 is composed of a single line segment connecting the pair of farthest points (extremes of the line segments) in C. Given the polyline P_{n-1}, call s the line segment in (a pair belonging to) C that is at the maximum distance from its (closest) corresponding line segment $\bar{s}$ in P_{n-1}. If the distance $d(s, \bar{s})$ is less than the acceptable error FUSIONTOLERANCE (set to $15\,\mathrm{mm}$ in our experiments), then P_{n-1} is the final approximation P. Otherwise, s is inserted in P_{n-1} to substitute $\bar{s}$ and s is connected to the two closest line segments in P_{n-1} to obtain the new polyline P_n.

The above algorithm is not guaranteed to terminate within a given time bound, because line segments in C can be considered an unpredictable number of times in building the approximating polyline. For this reason, we implemented a greedy version of the above (plain) algorithm in which a line segment s in (a pair in) C is considered only once for insertion in the polyline. The greedy method produces approximate polylines that are more “clean” than those produced by the plain method (Fig. 5). Moreover, given a matching chain C, the greedy version of the algorithm is guaranteed to terminate in $O(c)$ iterations, where c is the number of pairs in C. Note that, strictly speaking, the fusion of the scans presented in this section is not part of scan matching., as it is intended in literature. However, we use it in INTEGRATE to reduce the complexity (i.e., the number of line segments) of the resulting map.



(a) Matching chain (b) 1st iteration (c) 2nd iteration (d) 3rd iteration

4: An example of iterative construction of an approximating polyline, shown in black, for a matching chain, shown with two levels of gray

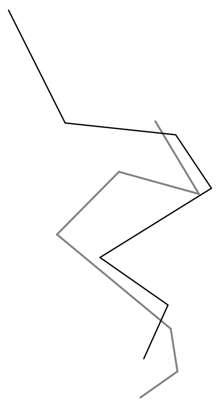
D. Analysis of Approximation Errors

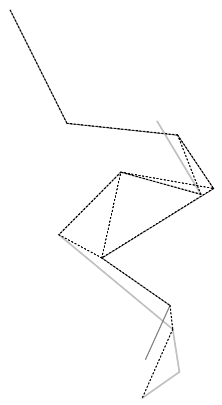
In this section, we present an analysis of the approximation errors introduced by the scan integration method described above. We assume that the points acquired by the laser range scanner are affected by an error of σ (typically, this value is around $1\,\mathrm{cm}$). This means that the real point in the environment lies within a circle centered in the point returned by the sensor and with radius σ.

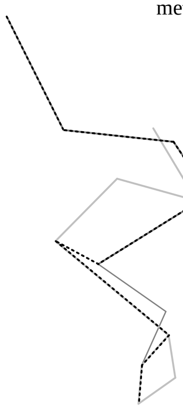
The points returned by the sensor are approximated with a set of line segments following the approach described in [39]. We operate in two steps: (1) the points are grouped into clusters and (2) a polyline is generated to fit the points in each cluster. In (1), we consider the acquired points in counterclockwise order and we group in the same cluster the consecutive points whose distance is less than a threshold τ (set to $20\,\mathrm{cm}$ in our experiments) from their successors. In (2) we approximate the points in a cluster by recursively building

Algorithm 5 Fusion of two scans

```
    Input: two scans, $S_1$ and $S_2^{\bar{t}}$, and a set of matching chains $\{C\}$ relative to $\bar{t}$
    Output: the map $S_{1,2}$
 1: $S_{1,2} \leftarrow$ empty
 2: add to $S_{1,2}$ the unmatched line segments of $S_1$ and $S_2^{\bar{t}}$
 3: for all $C \in \{C\}$ do
 4:     $P_0 \leftarrow$ line segment connecting the pair of farthest points (extremes of line segments) in $C$
 5:     $s \leftarrow$ line segment in (a pair in) $C$ that is at maximum distance from the line segment $\bar{s}$ in $P_0$
 6:     $n \leftarrow 0$
 7:     while $d(s, \bar{s}) \geqslant$ FUSIONTOLERANCE do
 8:         $n \leftarrow n + 1$
 9:         $P_n \leftarrow P_{n-1}$ after substituting $\bar{s}$ with $s$ and adding line segments that connect $s$ to the closest line segments in $P_{n-1}$
10:         $s \leftarrow$ line segment in (a pair in) $C$ that is at maximum distance from its closest line segment $\bar{s}$ in $P_n$
11:     end while
12:     add the polyline $P_n$ to $S_{1,2}$
13: end for
```


(a) Matching chain


(b) Polyline with plain method


(c) Polyline with greedy method

5: A matching chain (in black and gray) and the resulting polyline (dotted line)

a polyline: initially it connects the first and the last point in the cluster, then the farthest point from the current polyline becomes a new endpoint of the polyline; the process continues until all points in the cluster are within a distance κ (set to $25\,\mathrm{mm}$) from the polyline. Hence, the line segments in a scan approximate the points perceived by the laser range scanner. The polyline generation presented above resembles that of [40] that builds clusters on the basis of angles instead of distances. (Note that these polylines approximate perceived points in the post-processing of scan acquisition and are different from the polylines approximating matching chains in the fusion of scans.)

The clustering of points does not introduce any approximation error. The threshold of the clustering algorithm influences the number of polylines that are created but not the precision with which the points are approximated by these polylines. Since, in our experiments, the maximum range of the laser sensor has been set to $8\,\mathrm{m}$ and its angular resolution to $1°$, two consecutive points at the end of range of the sensor are separated by about $14\,\mathrm{cm}$ (obtained from $8\,\mathrm{m} \times \sin(1°)$). Hence, the clustering threshold τ has been set to $20\,\mathrm{cm}$ in order to keep in the same cluster two consecutive points at the end of the range. Once the points have been separated in clusters, the generation of the polyline approximating the points in a cluster introduces, by definition, a maximum (worst-case) error of κ.

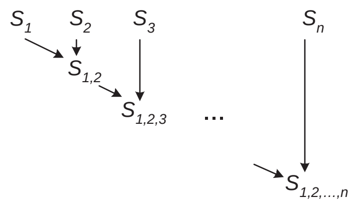
During scan integration, the fusion (see Section III-C) of the line segments of two scans introduces other approximation errors. More precisely, given a matching chain C, the maximum (worst-case) distance between a line segment of the resulting polyline and a line segment belonging to a pair in C (namely, a line segment of S_1 or $S_2^{\bar{t}}$) is FUSIONTOLERANCE in the plain version of the fusion algorithm. Our greedy implementation introduces a larger approximation error that has been experimentally evaluated (on a sample of scans from Section V) to be almost always less than $40\,\mathrm{mm}$.

Globally, the approximation error introduced by our scan integration approach is, in the *worst case*, $\kappa + 40\,\mathrm{mm}$, given that the points returned by the sensor (on which the algorithms work) are affected by an error of σ.

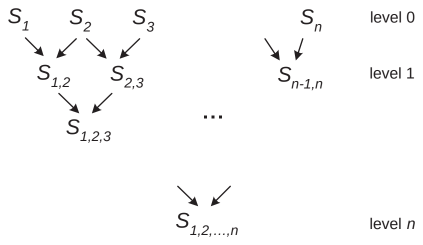
IV. Methods for Map Merging

The scan integration method discussed in the previous section produces a map $S_{1,2} = \mathrm{INTEGRATE}(S_1, S_2)$. The reference frame of $S_{1,2}$ coincides with the reference frame of S_1, since $\bar{t}$ is a transformation that brings the reference frame of S_2 in the reference frame of S_1. The main advantage of INTEGRATE is that, since it is not based on information about the relative position of S_1 and S_2 and it works with collections of line segments, it is applicable indifferently to situations in which S_1 and S_2 are scans and to situations in which S_1 and S_2 are partial maps. Obviously, in this second case, the partial maps could contain a larger number of line segments and the computational time would be larger.

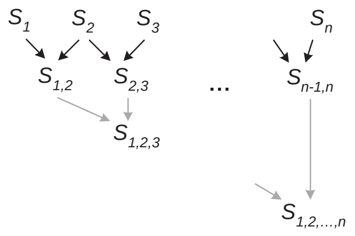
In this section, we describe three proposed methods (schematically shown in Fig. 6) for integrating a sequence $S_1, S_2, \ldots S_n$ of n partial maps by repeatedly calling INTEGRATE. (These methods have been introduced in [41].)



(a) Sequential method



(b) Tree method



(c) Pivot method

6: A schematic representation of the three map merging methods

Note that our contribution to the solution of the problem of integrating a sequence of partial maps is a step towards the solution of the more general (and complex) problem of integrating a set of partial maps. Some issues about this general problem are discussed in Section V-C.

A. Sequential Method

The simplest method is the *sequential method*. It operates as follows. The first two partial maps in the sequence are integrated, the obtained map then is grown by sequentially integrating the third partial map, and so on. Hence, S_1 is integrated with S_2 to obtain $S_{1,2}$, $S_{1,2}$ is integrated with S_3 to obtain $S_{1,2,3}$, and so on. Eventually, the final map $S_{1,2,\ldots,n}$ is constructed. In order to integrate n partial maps, the sequential method requires $n-1$ calls to INTEGRATE. A problem with the sequential method is that, as the process goes on, INTEGRATE is applied to a partial map that grows larger and larger (it contains more and more line segments). This will cause difficulties in the integration of S_i with large i, since S_i could match with different parts of the larger map $S_{1,2,\ldots,i-1}$.

B. Tree Method

To overcome the above problem, the integration of a small partial map with a large partial map should be avoided. This is the idea underlying the *tree method*, which works as follows. Each partial map of the initial sequence is integrated with the successive partial map of the sequence to obtain a new sequence $S_{1,2}$, $S_{2,3}$, $\ldots$, $S_{n-1,n}$ of $n-1$ partial maps. Then, each partial map of this new sequence is integrated with the successive one to obtain a new sequence $S_{1,2,3}$, $S_{2,3,4}$, $\ldots$, $S_{n-2,n-1,n}$ of $n-2$ partial maps. The process continues until a single final map $S_{1,2,\ldots,n}$ is produced.

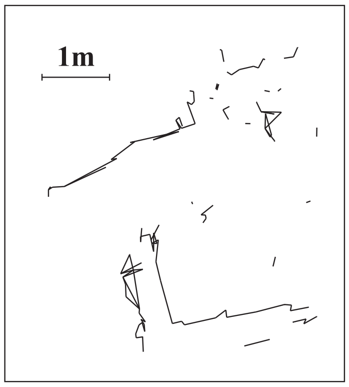
The tree method always integrates partial maps of similar size, since they approximately contain the same number of line segments. The number of calls to INTEGRATE required by the tree method to integrate a sequence of n partial maps is $n(n-1)/2$. Note also that, while it is quite obvious that the sequential method can be applied online (i.e., while the robot is moving), the most natural implementation of the tree method is offline, since it is not straightforward to devise an online algorithm for the tree method that requires constant time, as n grows, to update the tree (some results about online implementation are reported in Section V-B).

To speed up the tree method we have developed a heuristic that, given a sequence of partial maps at any level of the tree (let us suppose at level 0 for simplicity), attempts to integrate the partial maps S_i and S_{i+2}; if the integration succeeds (for example, a success can be experimentally determined by calculating if the value of the best transformation returned by INTEGRATE is above a threshold), the final result $S_{i,i+2}$ represents the same map that would have been obtained with three integrations: S_i with S_{i+1} to obtain $S_{i,i+1}$, S_{i+1} with S_{i+2} to obtain $S_{i+1,i+2}$, and $S_{i,i+1}$ with $S_{i+1,i+2}$ to obtain $S_{i,i+1,i+2}$. The number of partial maps in the new sequence is reduced by one unit, because $S_{i,i+2}$ substitutes both $S_{i,i+1}$ and $S_{i+1,i+2}$. This heuristic finds its natural applicability when the partial maps S_i and S_{i+2} are already the result of a number of integrations performed by the tree method and their common part is significant. For example, in the sequence produced at the level 3 of the tree technique the first ($S_{1,2,3,4}$) and the third ($S_{3,4,5,6}$) partial maps have a significant common part, since approximately half of the two partial maps overlaps. This improves the robustness of the method, since corresponding angles are likely to be found in the two partial maps.

A problem with the tree method is caused by the presence of "spurious" line segments in the integrated maps, namely line segments that correspond to the same part of the real environment but that are not fused together with the procedure of Section III-C, for example because their alignment is imprecise (Fig. 7). This problem is exacerbated in the tree method since the same parts of the partial maps are repeatedly fused together and errors accumulate.

C. Pivot Method

To avoid the problems of the sequential and tree methods, we devised the *pivot method* that combines the best features of the two above methods. This method starts as the tree method and constructs a sequence $S_{1,2}$, $S_{2,3}$, $\ldots$, $S_{n-1,n}$ of $n-1$ partial maps starting from the initial sequence. At this point, we note that S_2 is part of both $S_{1,2}$ and $S_{2,3}$ and that the transformation $\bar{t}_{1,2}$ used to integrate S_1 and S_2 provides the position and orientation of the reference frame of S_2 in the reference frame of $S_{1,2}$. It is therefore possible to transform $S_{2,3}$ according to $\bar{t}_{1,2}$ and fuse the line segments of the partial



7: Spurious line segments that have not been fused together in the final map

maps $S_{1,2}$ and $S_{2,3}^{\bar{t}_{1,2}}$ to obtain $S_{1,2,3}$. In a similar way, $S_{1,2,3,4}$ can be obtained from $S_{1,2,3}$ and $S_{3,4}$ by applying to the latter the transformations $\bar{t}_{2,3}$ and $\bar{t}_{1,2}$ and fusing the line segments of $S_{1,2,3}$ and $S_{3,4}^{\bar{t}_{2,3}\bar{t}_{1,2}}$. Iterating this process, from the sequence $S_{1,2}$, $S_{2,3}$, …, $S_{n-1,n}$ the final map $S_{1,2,\ldots,n}$ is obtained.

The pivot method integrates partial maps of the same size, like the tree method, and requires $n-1$ calls to INTEGRATE, like the sequential method. (In addition it requires $n-2$ executions of the not-so-expensive step 3 of INTEGRATE, see Algorithm 1.) Integrating the line segments of two scans only once, the pivot method reduces the problem of spurious line segments. The pivot method is also naturally implementable in an online system. The problem of spurious line segments is reduced but not completely eliminated by the pivot method; a way to further reduce this problem is to fuse not $S_{1,2}$ and $S_{2,3}^{\bar{t}_{1,2}}$, but $S_{1,2}$ and $S_{3}^{\bar{t}_{1,3}}$, where $\bar{t}_{1,3}$ is the composition of $\bar{t}_{1,2}$ and $\bar{t}_{2,3}$. The pseudo-code of the algorithm for this pivot method is reported as Algorithm 6.

Algorithm 6 Pivot method for map merging

```
   Input: a sequence of scans, $S_1, S_2, \ldots, S_n$
   Output: a final map $S_{1,2,\ldots,n}$
1: $S_{1,2} \leftarrow$ INTEGRATE$(S_1, S_2)$ and store $\bar{t}_{1,2}$
2: for i = 3,4,...,n do
3:     $S_{i-1,i} \leftarrow$ INTEGRATE$(S_{i-1}, S_i)$ and store $\bar{t}_{i-1,i}$
4:     $\bar{t}_{1,i} \leftarrow$ compose $\bar{t}_{1,i-1}$ and $\bar{t}_{i-1,i}$
5:     $S_i^{\bar{t}_{1,i}} \leftarrow$ transform $S_i$ according to $\bar{t}_{1,i}$
6:     $S_{1,2,\ldots,i} \leftarrow$ apply the fusion procedure (Algorithm 5)
       to $S_{1,2,\ldots,i-1}$ and $S_i^{\bar{t}_{1,i}}$
7: end for
```

V. Experimental Results

The experimental validation of our methods has been done both with data collected in our laboratory (Sections V-A and V-B) and with data publicly available on the Internet (Section V-C). In our laboratory, we used a SICK LMS 200 laser range scanner (mounted on a Robuter mobile platform at a height of approximatively $50\,\mathrm{cm}$) to acquire a sequence of distance measurements along directions separated by a programmable angle ($1°$, in our case) sweeping $180°$. The result of a sensing operation is thus a set of points expressed in polar coordinates, with the origin of the coordinate frame in the sensor itself. These points are approximated by line segments, as described in Section III-D.

For the experiments of Sections V-A and V-B we acquired 31 scans (Table I). The scans have been acquired in different environments (forming a loop about $40\,\mathrm{m}$ long) by driving the robot manually and without recording any odometric information. We started from a laboratory, a very scattered environment, then we crossed a narrow hallway with rectilinear walls to enter a department hall, a large open space with long perpendicular walls, and finally we closed the loop re-entering the laboratory (see the dashed path in Fig. 12). The experiments have been designed to include a variety of cases and to stress the algorithms we propose. The correctness of the integrations has been determined by visually evaluating the maps with respect to the real environment. The displacements (both translational and rotational) between the scans are significant. The translational displacements are between $34\,\mathrm{cm}$ and $2.8\,\mathrm{m}$, with an average of $1.2\,\mathrm{m}$. The rotational displacements are between $0.0035\,\mathrm{rad}$ and $1.73\,\mathrm{rad}$, with an average of $0.33\,\mathrm{rad}$. (These values have been derived from the automated matching of the scans performed by our method.) These displacement values are much more than those usually reported in literature. We note that large displacements allow the mapping process to quickly cover the environments with few steps.

The programs have been coded in ANSI C++ employing the LEDA libraries 4.2 and have been run on a $1\,\mathrm{GHz}$ Pentium III processor with Linux SuSe 8.0. We stress that our approach is independent of the robots used to acquire the scan data, as shown in Section V-C; thus can be naturally applied in a multirobot context, provided that the scans are taken at the same height.

A. Scan Integration Experiments

For every pair of consecutive scans acquired in our laboratory, we tested the basic method for scan integration and the three heuristics, sometimes modifying the values of the parameters. SegmentLengthPercentage ranged from 2 (for scans with long line segments) to 40 (for scans with short line segments). SegmentLengthRefuse ranged from $80\,\mathrm{cm}$ (for scans with short line segments) to $140\,\mathrm{cm}$ (for scans with long line segments).

In general, our experimental results demonstrate that the proposed scan integration method performs well (Table II), but not all the pairs can be integrated. 28 pairs of scans out of 31 possible pairs (S_{31} is integrated with S_1) have been correctly matched with at least one of the heuristics presented in Section III (last row of Table II). Unsurprisingly, the histogram-based heuristic worked well with scans containing long and perpendicular line segments, like those taken in the hallway and in the hall. The heuristic that considers consecutive line segments seems to work well in all three kinds of environment, even if sometimes it needs some parameter adjustments.

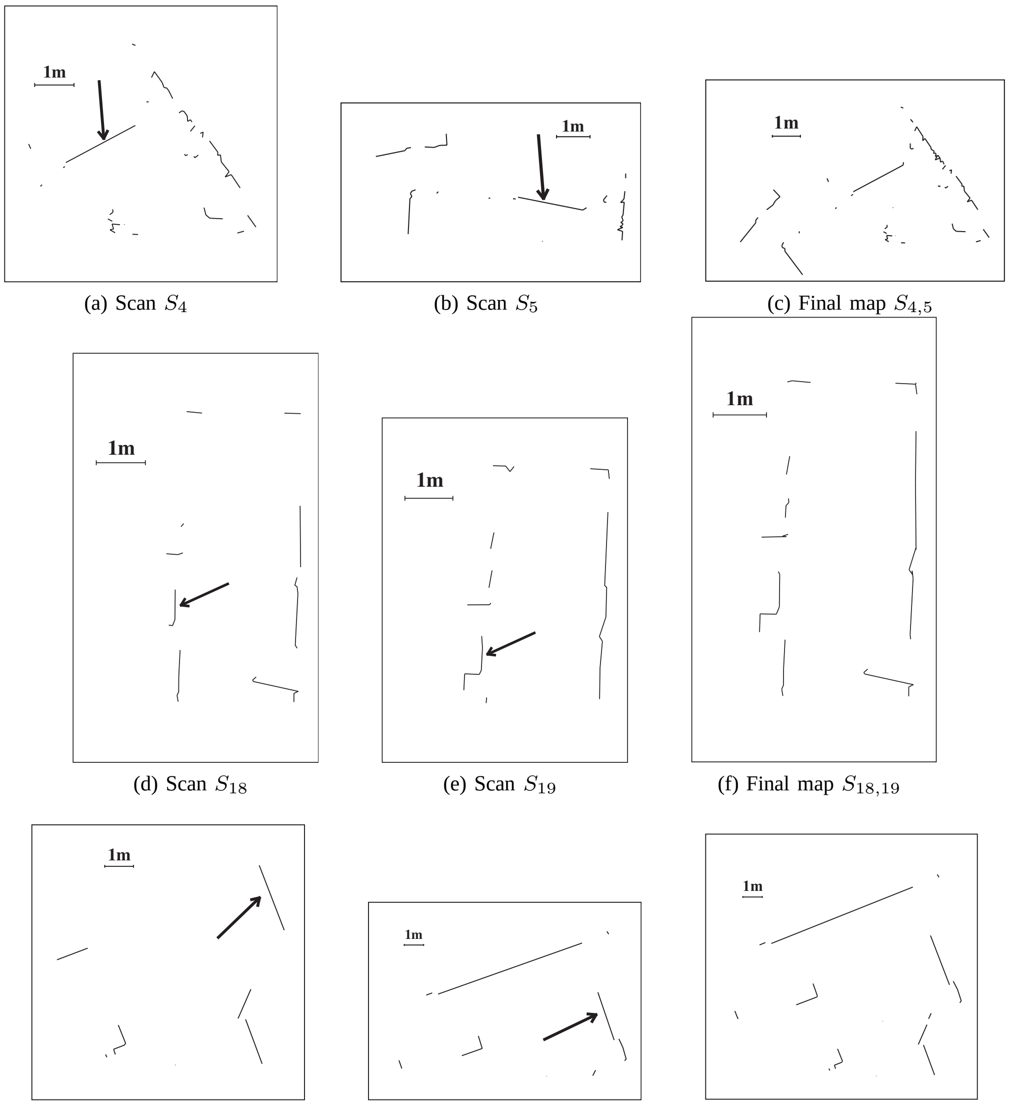
Table III shows the results obtained by integrating three interesting pairs of scans (see also Fig. 8). S_4 and S_5 were

I: Scans acquired in our laboratory (line segment lengths are in mm)

Environment	Scans	Avg number of line segments	Avg length of line segments
Laboratory	$S_1 - S_9, S_{30} - S_{31}$	39.5	212.3
Hallway	$S_{10} - S_{24}$	19.3	366.3
Hall	$S_{25} - S_{29}$	15.6	607.0
Total	$S_1 - S_{31}$	25.9	350.4

II: Scan integration results over the 31 scans acquired in our laboratory

Heuristic used	Number (%) Successes	Number (%) Failures
All transformations	13 (41.9%)	18 (58.1%)
Consecutive line segments	21 (67.7%)	10 (32.3%)
Random line segments	10 (32.2%)	21 (67.8%)
Histogram	9 (29%)	22 (71%)
Using the best heuristic	28 (90%)	3 (10%)



(a) Scan S_4 (b) Scan S_5 (c) Final map $S_{4,5}$

(d) Scan S_{18} (e) Scan S_{19} (f) Final map $S_{18,19}$

(g) Scan S_{25} (h) Scan S_{26} (i) Final map $S_{25,26}$

8: Pairs of scans and resulting final maps (the arrows show line segments corresponding to the same object in the environment)

taken inside the laboratory: they contain a large number of short line segments since the environment is highly scattered. The heuristic that works better is that based on consecutive line segments: it was able to find a good transformation evaluating only two transformations. On the other hand, the evaluation of all the possible transformations is infeasible (over $40,000$ matches to evaluate!). S_{18} and S_{19} were taken along the hallway: they contain fewer line segments than the previous scans and are characterized by long rectilinear line segments. Even in this case, evaluating all the transformations is expensive, while the consecutive line segments heuristic performs well. S_{25} and S_{26} were taken in the hall: they contain only few line segments since the environment is characterized by long rectilinear and perpendicular walls. All the heuristics perform well in this case because, starting from a small number of line segments, there are only few transformations that are easy evaluated.

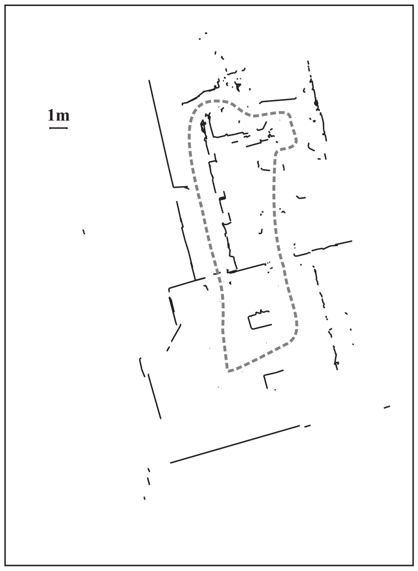
For scan pairs $S_1 - S_2$ and $S_2 - S_3$ our method was not able to find the correct transformation. As shown in Fig. 9, these scans contain many short line segments representing scattered small objects (chairs, tables, robots, and boxes). It is almost impossible, even for a human being, to find the correct match between these scans without any prior information about their relative positions. Similar problems emerged in the hall. For example, Fig. 10 shows scans S_{27} and S_{28}, where the second one has been taken after rotating the robot of about 100 degrees. Since the environment is large and has only few objects that can be used as reference, a drastic change of the field of view eliminates any common reference between scans, thus automatic matching is impossible.

We now discuss the role of the parameters that influence the performance of our scan integration method. POINTDISTANCETOLERANCE affects the matching value of two line segments and the corresponding transformation. In the same way, large values for SEGMENTDISTANCETHRESHOLD make line segments that do not represent the same object in the environment to match; small values reduce the number of matching line segments thus making the method more sensitive to measurement errors. Large values of ANGLEDIFFERENCETOLERANCE facilitate the search of the best transformation by allowing many possible transformations to be considered, but their evaluation requires more time.

B. Map Merging Experiments

The sequence of scans we considered for validating our map merging methods is composed of 29 scans $S_3, S_4, \ldots, S_{31}$. We have excluded the three initial scans from the sequence acquired in our laboratory because, as discussed in Section V-A, they could not be integrated. Moreover, in order to close the loop and complete the experiments, scans from S_{27} to S_{29} (Fig. 10) were manually integrated. In the following, we discuss the integration of this sequence of scans done offline to test and compare all the three methods presented above.

Fig. 11 shows the final map (composed of 278 line segments) obtained with the sequential method. The sequential method could not integrate all the scans in order to close the loop: the method suddenly failed when we tried to integrate S_{23}. It is evident that S_{23} has only a few short line segments in common with the rest of the map. Furthermore, as already discussed, when the global map grows during the sequential integration, the scan matching becomes computationally very difficult because the large number of line segments requires a high effort for evaluating the possible transformations. For example, the integration of S_{19} (composed of 28 line segments) with $S_{3,4,\ldots,18}$ (composed of 247 line segments) takes $5.17\,\mathrm{s}$.



12: The final map obtained with the tree method (with the dashed path followed by the robot)

IV: Computing time (in s) for online map merging

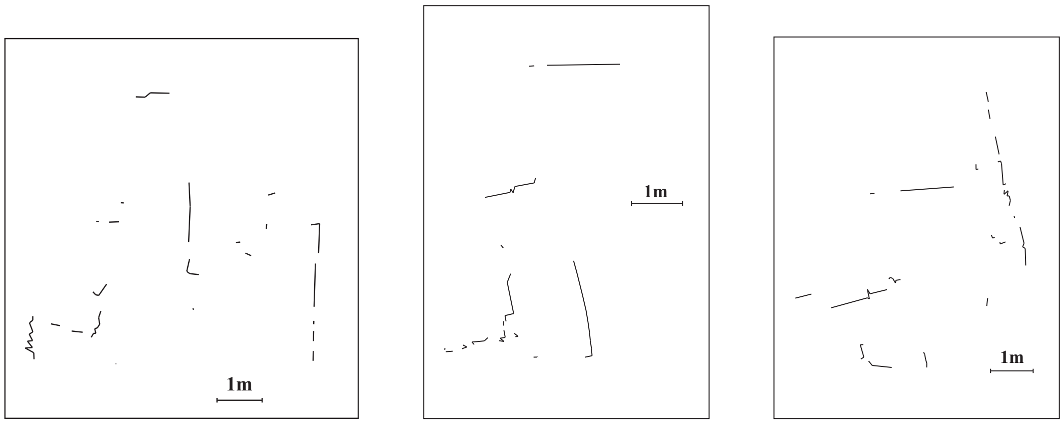
Newly acquired scan	Sequential	Tree	Pivot
S_4	0.3	0.3	0.3
S_5	0.3	0.8	0.3
S_6	0.6	2.8	0.6
S_7	0.8	6.5	0.7
S_8	0.7	14.3	0.7
S_9	0.7	26.6	0.7

Fig. 12 shows the final map (composed of 519 line segments) obtained with the tree method. We applied the standard tree method until level 3 of the tree, then we applied the heuristic presented in Section IV-B to speed up the process. As we went down in the tree, the size of the maps grew larger and larger and the execution of INTEGRATE slowed down. For example, the integration of two partial maps (composed of 108 and 103 line segments) at level 3 of the tree requires $12.8\,\mathrm{s}$. Furthermore, as already noted, when we integrate large-sized maps with many redundant spurious line segments that represent the same part of the environment, the resulting maps are noisier because of the error introduced when attempting to integrate maps with many overlapping line segments.

Fig. 13 shows the final maps obtained with the pivot method. The map on the left is composed of 441 line segments and has been built by fusing the partial map $S_{i-1,i}$ with $S_{i,i+1}^{\bar{t}_{i-1,i}}$, while the map on the right is composed of 358 line segments and has been built by the optimized method that fuses the partial map $S_{i-1,i}$ with $S_{i+1}^{\bar{t}_{i-1,i+1}}$. The second map presents fewer spurious line segments and appears more “clean”.

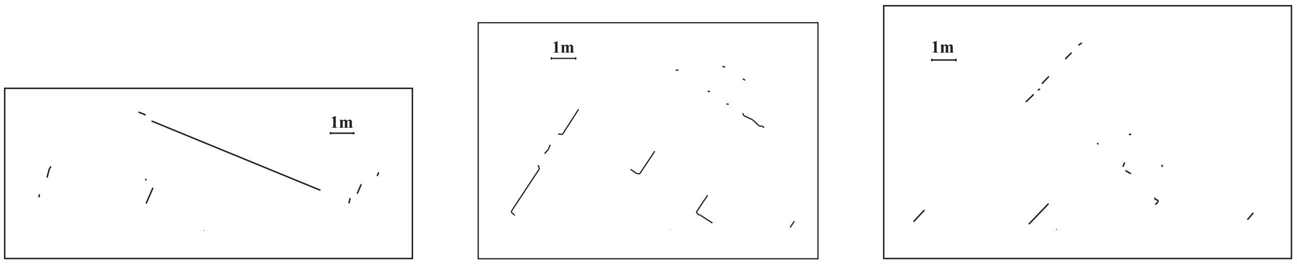
We have preliminary tested the performance of the online implementation of the map merging methods, considering the sub-sequence of scans $S_3, S_4, \ldots, S_9$ and the consecutive line segment heuristic. Results are shown in Table IV in which the time needed to integrate a newly acquired scan in the previous global map is reported. The sequential and the pivot methods are the best options for online implementation.

Given the nature of our approach, there is no *a priori*



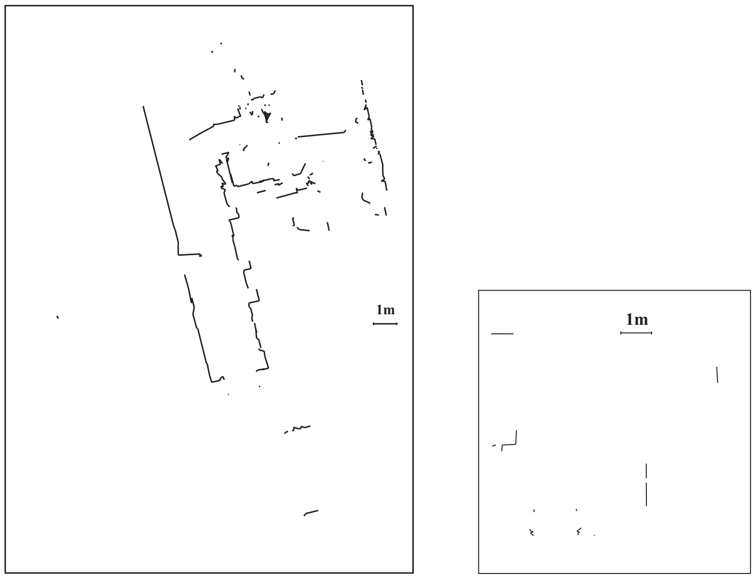
(a) Scan S_1 (b) Scan S_2 (c) Scan S_3

9: Scans taken in the lab entrance



(a) Scan S_{27} (b) Scan S_{28} (c) Scan S_{29}

10: Scans taken in the hall

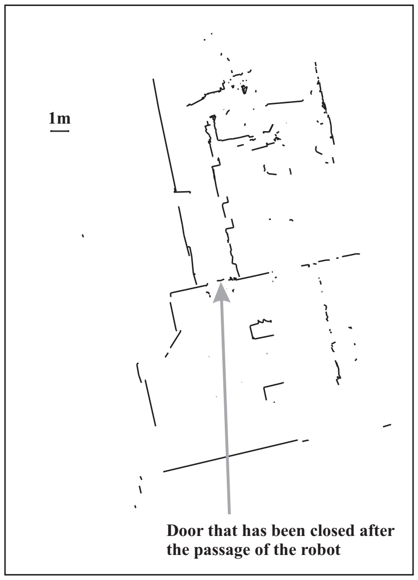


(a) Sequential method (b) Scan S_{23}

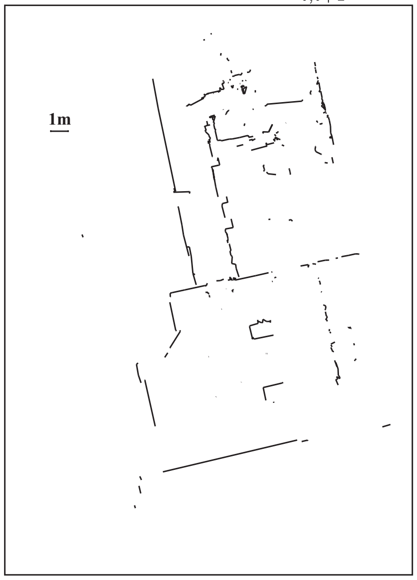
11: The final map obtained with the sequential method for scans S_1 to S_{22} and scan S_{23}

III: Interesting scan integration examples (times are in s)

Scans	S_4	S_5	S_{18}	S_{19}	S_{25}	S_{26}
# of line segments	47	36	24	24	10	12
	Time	# tried	Time	# tried	Time	# tried
All transformations	936	41,260	32	3,096	0.38	231
Consecutive line segments	1.25	2	0.73	27	0.13	4
Random line segments	7.69	∼20,000	2.51	∼20,000	0.78	∼20,000
Histogram	3.29	73	1.97	192	0.15	32



(a) Fusion of $S_{i-1,i}$ with $S_{i,i+1}^{\bar{t}_{i-1,i}}$



(b) Fusion of $S_{i-1,i}$ with $S_{i+1}^{\bar{t}_{i-1,i+1}}$

13: The final maps obtained with the pivot method

guarantee that the final map is consistent. Actually, this is a critical issue for all map merging methods. In our case, the consistency of the final map can be ensured by the user who can validate each integration performed by the method. A more flexible solution is to let the user set a threshold for the value of the best transformation in INTEGRATE. An integration is considered to be valid only when its value (i.e., the value of its associated best transformation, see Section III-B) is above the threshold. By setting the values of the threshold smaller or larger, the user can decide to be more or less confident with the results produced by our method.

C. Further Experimental Results

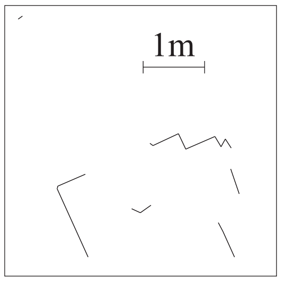
To further validate our approach and to show that it works also with different data, we applied it to the stanford-gates1 data set available in the Robotics Data Set Repository (Radish) [42] (thanks to Brian Gerkey for providing these data). This data set is a 30-minute tour through the first floor of the Stanford's Gates Computer Science Building. The robot used to collect the data is a Pioneer 2DX with a forward-pointing SICK LMS 200. The laser was running at high speed ($75\,\mathrm{Hz}$ scans) in the $10\,\mathrm{mm}$, 1° mode. The data set includes both laser data and odometry data. We considered only laser data (about $115,000$ laser scans!). For each scan of the data set, we approximate the points acquired by the laser range scanner by line segments, as described in Section III-D. We call scans S_x, where x is the time (in seconds) at which a scan has been acquired, according to the timestamps reported in the data set. To obtain good experimental results, we set some parameters to values different from those used in the previous sections: POINTDISTANCETOLERANCE has been set to $10\,\mathrm{mm}$ and SEGMENTDISTANCETHRESHOLD to 10.

The first set of experiments we performed with the stanford-gates1 data set is devoted to show that our method can always find the correct integration between two scans, provided that the two scans are taken close enough. For example, scans S_{518} and S_{522} (taken $4\,\mathrm{s}$ apart) are not correctly integrated with our method but, when considering also S_{520}, our method correctly integrates S_{518} with S_{520} and the result of this integration with S_{522} (Fig. 14). The same happens for scans S_{316}, S_{318}, and S_{320} (Fig. 15).

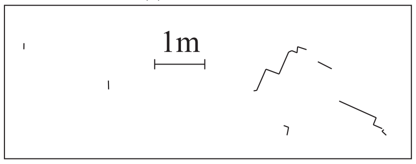
To corroborate the results of Table II and to compare the heuristics for scan integration of Section III-A in a different environment, we applied them to 15 pairs of scans (the scans of each pair has been taken at $4\,\mathrm{s}$ from each other) randomly selected from the stanford-gates1 data set. Results are reported in Table V. An interesting future research direction could be the automatic identification, given an environment, of the best heuristic.

V: Scan integration results over 15 pairs of scans of the stanford-gates1 data set

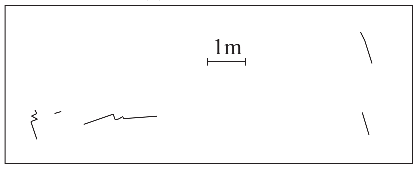
Heuristic used	Number (%) Successes	Number (%) Failures
All transformations	13 (86.7%)	2 (13.3%)
Consecutive line segments	8 (53.3%)	7 (46.7%)
Random line segments	2 (13.3%)	13 (86.7%)
Histogram	7 (46.7%)	8 (53.3%)
Using the best heuristic	14 (93.3%)	1 (0.7%)



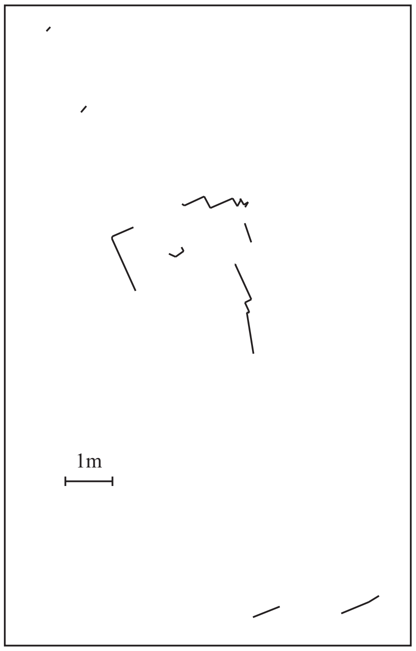
(a) Scan S_{518}



(b) Scan S_{520}

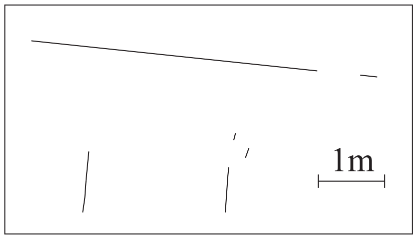


(c) Scan S_{522}

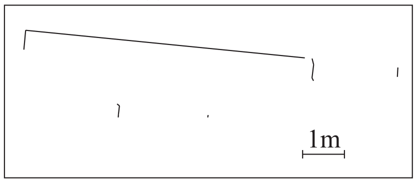


(d) Resulting map

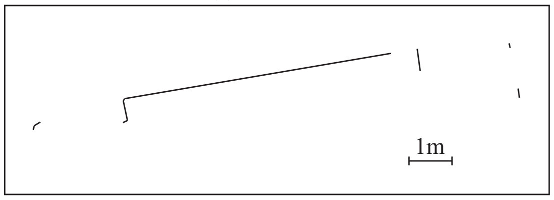
14: Scans and the map resulting from their integration



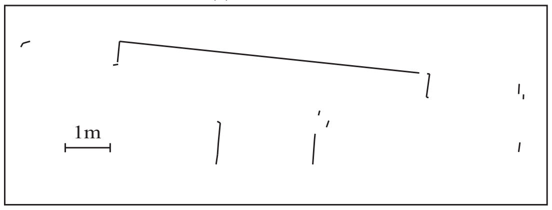
(a) Scan S_{316}



(b) Scan S_{318}



(c) Scan S_{320}

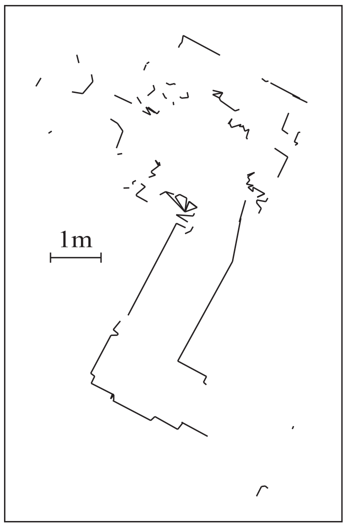


(d) Resulting map

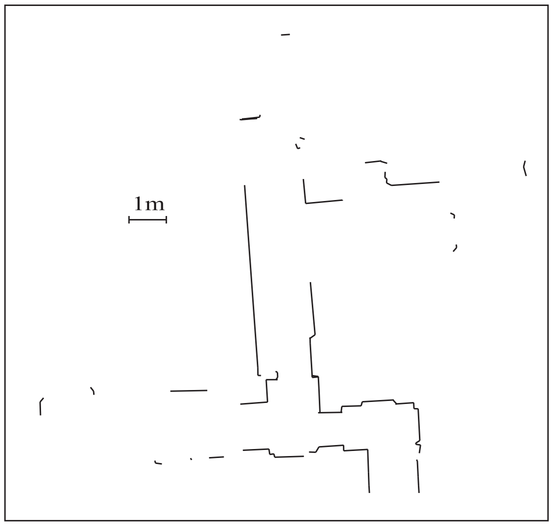
15: Scans and the map resulting from their integration

We also analyzed the robustness of our method with respect to variations of the parameter values. To this end, we considered three randomly selected pairs of scans of stanford-gates1 data set and we applied our scan integration method (with the consecutive line segment heuristic) varying the values of POINTDISTANCETOLERANCE, ANGLEDIFFERENCETOLERANCE, and SEGMENTDISTANCETHRESHOLD. The method has been able to correctly integrate the pairs when the above parameters had values within $3\,\mathrm{mm}$ and $13\,\mathrm{mm}$ for POINTDISTANCETOLERANCE, $0.16\,\mathrm{rad}$ and $1.53\,\mathrm{rad}$ for ANGLEDIFFERENCETOLERANCE, and 8 and 131 for SEGMENTDISTANCETHRESHOLD.

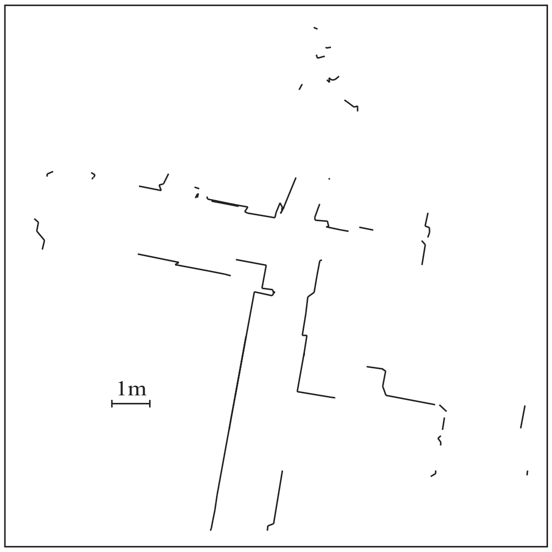
The last experiment we performed with the stanford-gates1 data set is a simulation of a realistic multirobot setting. We assumed that four mobile robots had individually acquired four local maps of the environment (Fig. 16). According to the two-level multirobot scenario depicted in Section I, we built these



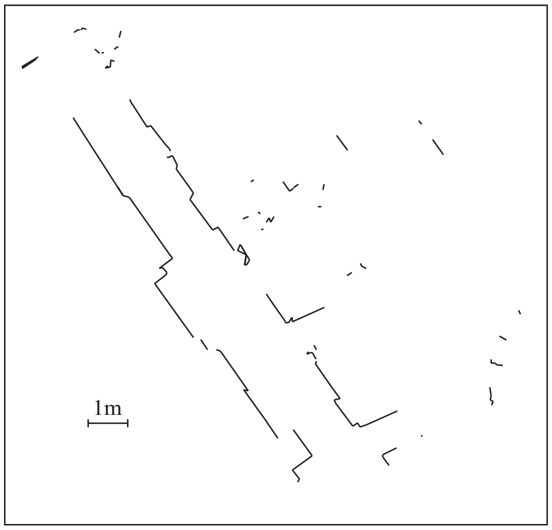
(a) Local map L_1



(b) Local map L_2



(c) Local map L_3



(d) Local map L_4

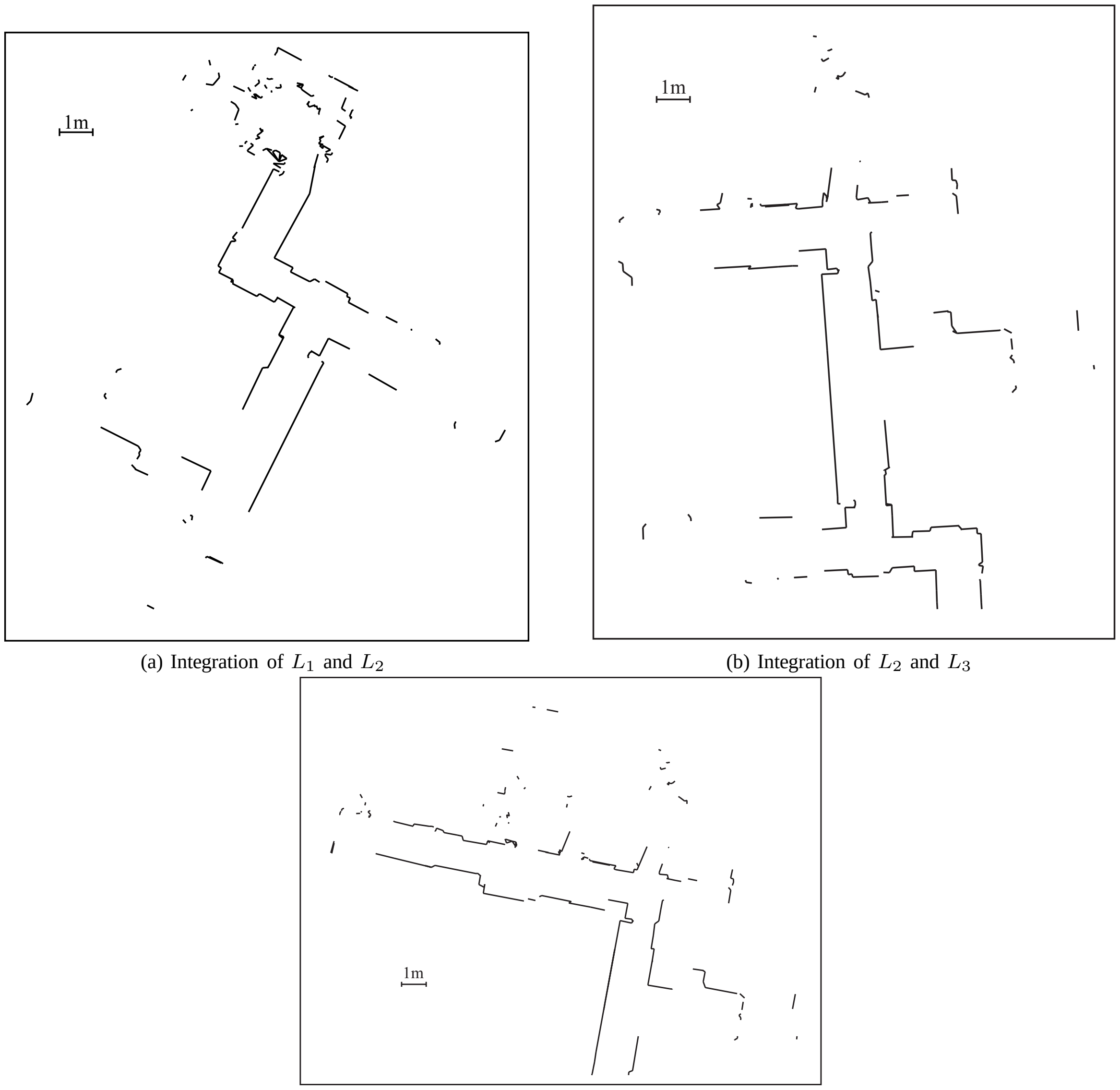
16: Local maps that could have been acquired by four different robots

local maps, called L_1, L_2, L_3, and L_4, by integrating four sequences of 10 scans (taken at $4\,\mathrm{s}$ from each other). Each local map L_i has two scans in common with the local maps L_{i-1} and L_{i+1} (when they exist). The total length of the line segments in each local map is about $35\,\mathrm{m}$. Our scan integration method has been able to integrate correctly the pairs of local maps, as shown in Fig. 17. The time required to integrate (with the heuristic that considers consecutive line segments) two local maps is about $20\,\mathrm{s}$. Note that we tried to integrate *all* the pairs of local maps. The correct matches have a best transformation value of about $10\,\mathrm{m}$, while the wrong matches have a best transformation value of about $5\,\mathrm{m}$. For example, the best transformation value between L_3 and L_4 is $14.8\,\mathrm{m}$, while the best transformation value between L_1 and L_4 is $4.3\,\mathrm{m}$.

VI. Conclusions and Future Work

In this paper we have presented an approach for integrating pairs of partial maps composed of line segments and for merging a sequence of partial maps in order to build a global map. Our method works without any information about the relative poses of the partial maps but relies exclusively on their geometric features. The advantage of using geometric features is that the representation based on line segments is very compact and the maps produced are easy to use. This is the major aspect which distinguishes our approach from other robot mapping methods reported in the literature. The methods presented in this paper provide an elegant solution to the problem of multirobot mapping since they are independent from where and by which robot the partial maps have been acquired. Experimental results validate the effectiveness of the approach for indoor environments.

In future research we plan on generalizing these methods following the preliminary results of Section V-C, to cases where the order in which the partial maps have to be integrated is not known. This would happen, for instance, when maps are created by different robots since we cannot assume the order in which the merging will be done is the same as the order in



(a) Integration of L_1 and L_2

(b) Integration of L_2 and L_3

(c) Integration of L_3 and L_4

17: The result of the integration of local maps

which they have been acquired. Note that the problem is not as severe as it might appear unless the number of robots is very large. We will also explore how adding positional information will affect the performance of the methods and examine how sensitive they are to pose errors.

Acknowledgments

The authors would like to thank Jean-Claude Latombe for his generous hospitality at Stanford University where this research was started, Héctor Gonzáles-Baños for sharing his programs and expertise with collecting laser range scan data, Brian Gerkey for providing the Stanford data set, Paolo Mazzoni and Emanuele Ziglioli for the initial implementation of the fusion algorithm, Daniele Menotti for contributions to error analysis, and Igor Giussani for some experimental results. Francesco Amigoni was partially supported by a Fulbright fellowship and by "Progetto Giovani Ricercatori" 1999 grant, Maria Gini by NSF under grants EIA-0224363 and EIA-0324864.

References

[1] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2003.

[2] R. Chatila and J.-. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1985.

[3] E. Brunskill and N. Roy, "SLAM using incremental probabilistic PCA and dimensionality reduction," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2005.

[4] S. Engelson and D. McDermott, "Error correction in mobile robot map learning," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1992.

[5] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. Papanikolopoulos, "Performance of a distributed robotic system using shared communications channels," *IEEE T ROBOTIC AUTOM*, vol. 22, no. 5, pp. 713–727, Oct. 2002.

[6] P. E. Rybski, F. Zacharias, J.-F. Lett, O. Masoud, M. Gini, and N. Papanikolopoulos, "Using visual features to build topological maps of indoor environments," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2003.

[7] W. Grimson, *Object recognition by computer: the role of geometric constraints*. The MIT Press, 1990.

[8] W. Burgard, M. Moors, and F. Schneider, "Collaborative exploration of unknown environments with teams of mobile robots," in *Advances in Plan-Based Control of Robotic Agents*. Springer-Verlag, 2002, pp. 52–70.

[9] J. Ko, B. Stewart, D. Fox, and K. Konolige, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2003, pp. 3232–3238.

[10] J. Fenwick, P. Newman, and J. Leonard, "Cooperative concurrent mapping and localization," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2002, pp. 1810–1817.

[11] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE T ROBOTIC AUTOM*, vol. 12, no. 6, pp. 869–880, 1996.

[12] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE T ROBOTIC AUTOM*, vol. 17, no. 3, pp. 229–241, 2001.

[13] J. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE T ROBOTIC AUTOM*, vol. 17, no. 3, pp. 242–257, 2001.

[14] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proc. of the Nat'l Conf. on Artificial Intelligence*, 2000, pp. 852–858.

[15] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *MACH LEARN and AUTON ROBOT (joint issue)*, vol. 31, no. 5, pp. 1–25, 1998.

[16] ——, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2000, pp. 321–328.

[17] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *INT J ROBOT RES*, 2001.

[18] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, 2004.

[19] A. Howard, "Multi-robot mapping using manifold representations," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2004, pp. 4198–4203.

[20] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 7–40, 2001.

[21] D. Hähnel, D. Fox, W. Burgard, and S. Thrun, "A highly efficient fast-SLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," in *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2003.

[22] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, no. 2, pp. 61–74, 1988.

[23] A. Doucet, J. F. G. de Freitas, K. Murphy, and S. Russell, "Rao-Blackwellized particle filtering for dynamic Bayesian networks," in *Proc. Conf. on Uncertainty in Artificial Intelligence*, 2000.

[24] K. Konolige, D. Fox, B. L. J. Ko, and B. Stewart, "Map merging for distributed robot navigation," in *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2003.

[25] J.-S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *IEEE Int'l Symp. on Computational Intelligence in Robotics and Automation*, 1999.

[26] F. Lu and E. Milios, "Gloablly consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[27] ——, "Robot pose estimation in unknown environments by matching 2D range scans," *J INTELL ROBOT SYST*, vol. 18, no. 3, pp. 249–275, 1998.

[28] I. Cox, "Blanche: An experiment in guidance and navigation of an autonomous robot vehicle," *IEEE T ROBOTIC AUTOM*, vol. 7, no. 2, pp. 193–204, 1991.

[29] J. S. Gutmann and C. Schlegel, "Amos: Comparison of scan matching approaches for self-localization in indoor environments," in *Proc. of the Euromicro Workshop on Advanced Mobile Robots*, 1996, pp. 61–67.

[30] G. Weiss, C. Wetzler, and E. V. Puttkamer, "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans," in *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 1994, pp. 12–16.

[31] T. Röfer, "Building consistent laser scan map," in *Proc. of the European Workshop On Advanced Mobile Robots*, 2001, pp. 83–90.

[32] L. Zhang and B. Ghosh, "Line segment based map building and localization using 2D laser rangefinder," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2000, pp. 2538–2543.

[33] B. Tovar, R. Murrieta-Cid, and C. Esteves, "Robot motion planning for map building," in *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2002, pp. 673–680.

[34] A. Martignoni III and W. Smart, "Localizing while mapping: A segment approach," in *Proc. of the Eighteen Nat'l Conf. on Artificial Intelligence*, 2002, pp. 959–960.

[35] T. Einsele, "Real-time self-localization in unknown indoor environments using a panorama laser range finder," in *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 1997, pp. 697–702.

[36] L. J. Latecki, R. Lakaemper, X. Sun, and D. Wolter, "Building polygonal maps from laser range data," in *Int'l Cognitive Robotics Workshop*, 2004.

[37] M. Tomono, "A scan matching method using euclidean invariant signature for global localization and map building," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2004, pp. 866–871.

[38] X. Zezhong, L. Jilin, and X. Zhiyu, "Scan matching based on CLS relationships," in *Proc. IEEE Int'l Conf. on Robotics, Intelligent Systems and Signal Processing*, 2003.

[39] H. H. Gonzáles-Baños and J. C. Latombe, "Navigation strategies for exploring indoor environments," *INT J ROBOT RES*, vol. 21, no. 10-11, pp. 829–848, 2002.

[40] J. Canou, G. Mourioux, C. Novales, and G. Poisson, "A local map building process for a reactive navigation of a mobile robot," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2004.

[41] F. Amigoni, S. Gasparini, and M. Gini, "Scan matching without odometry information," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 2004.

[42] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/